# A Template for Two or One-Column Vignettes

**First Author**[a] **and Second Author**[a,b]

[a]Institute of Smoke and Magic, University of Sometown, Sometown, XY, 12345; [b]Department of Neat Tricks, Whereever State University, Someplace, MC, 67890

This version was compiled on May 28, 2023

**Your abstract will be typeset here, and used by default a visually distinctive font. An abstract should explain to the general reader the major contributions of the article.**

one | two | optional | keywords | here

## 1. Introduction

We propose a deep-learning approach for cell classification in image data to overcome the limitations of traditional manual cell analysis methods for cell types. Cell analysis is crucial in various scientific domains such as biology, medicine, and drug discovery. However, manual methods are labour-intensive, time-consuming, and require more comprehensive feature representation and generalizability.

Our approach holds significance for pharmaceutical companies engaged in drug discovery and cell behaviour analysis. It aims to improve efficiency, productivity and provide reliable cell classification results. Consequently, this approach can lower costs and increase profits while improving human health.

Deep learning models offer distinct advantages beyond classical machine learning, including automated feature learning, improved generalisation capabilities, and streamlined analysis. These models eliminate the need for manual feature engineering by automatically learning relevant features from raw data. This allows for more comprehensive and expressive representations, enhancing the overall effectiveness of cell classification.

Our study aims to address two key research questions. Firstly, we seek to enhance the interpretability of deep learning models for cell classification, enabling researchers to understand the underlying patterns and features the models use. This understanding is essential for gaining insights into the decision-making process and establishing trust in the predictions made by the models. Secondly, we investigate the impact of variations in training data on the performance of different neural architectures. By systematically manipulating the training data and evaluating model performance, we can gain valuable insights into the robustness and adaptability of deep learning models for cell classification tasks.

## 2. Methodology

**2.1 Methodology overview.** Figure 1 illustrates the research process, including data collection, initial data analysis, data cleaning and preprocessing, model design, training and optimisation, as well as evaluation.

**2.2 Data collection.** We obtained the data for our study from 10X Genomics (2023), consisting of microscopy images of the mouse brain coronal section in the Tagged Image File Format (TIFF or TIF). The gene expression levels of each cell were categorised into 28 distinct clusters, identified by cluster IDs ranging from 1 to 28. Hence, we obtained a CSV file with columns for cell IDs and corresponding clusters.

We first establish a structured folder hierarchy to store the images, with each cluster assigned a dedicated folder for its corresponding images. This approach ensures efficient management and retrieval of the processed data.

To extract the necessary data, we utilised information on cell boundaries available in the dataset. Unlike our previous laboratory work, where we only used a random sample of 1000 images, we revisited the original dataset and extracted over 36,000 images, encompassing all cells available. We decide to fully harness the dataset's potential and maximise its utilisation for our analysis. We named each cell image extracted in the "Cell_{X}" format, where X represents the cell id, and saved it as a PNG file in the folder of its corresponding clusters.

**Fig. 1.** Project Schematic Overview - Four key phase of project ///// I couldn't find the full resolution image for the diagram so here's a frog

### 2.3 Initial Data Analysis.

```
library(dplyr)
library(ggplot2)
library(png)

## change according to file path
folder_path <- "cell_images"
clusters <- paste0("cluster_", 1:28)
df <- data.frame(cluster = character(),
      num_images = integer(),
      stringsAsFactors = FALSE)

for (cluster in clusters) {
  cluster_path <- file.path(folder_path, cluster)
  num_images <- length(list.files(cluster_path,
      pattern = "^cell_.*\\.png$"))
  df <- df %>% add_row(cluster = cluster,
      num_images = num_images)}

df <- df %>% arrange(num_images)

ggplot(df, aes(x = reorder(cluster, num_images),
      y = num_images)) +
  geom_bar(stat = "identity") +
  labs(x = "Cluster", y = "Number of Images") +
  theme(axis.text.x = element_text(angle = 90,
        vjust = 0.5, hjust=1)) +
  ggtitle("Number of Images in Each Cluster
        (sorted by ascending order)")

image_data <- data.frame(Width = numeric(),
      Height = numeric())
```

```
for (cluster_num in 1:28) {
  cluster_dir <- paste0("cell_images/cluster_",
        cluster_num)
  for (file_name in list.files(cluster_dir,
        pattern = "^cell_\\d+\\.png$",
        full.names = TRUE)) {
    img <- readPNG(file_name)
    width <- dim(img)[2]
    height <- dim(img)[1]
    image_data <- rbind(image_data,
        data.frame(Width = width, Height = height))
  }
}

ggplot(image_data, aes(x = Width, y = Height)) +
  geom_point(size = 0.5) +
  xlab("Width") +
  ylab("Height") +
  ggtitle("Image Dimensions EDA")
```

Figure 2 illustrates a notable trend in the dataset, showing a gradual rise in the number of images as we move from cluster 28 to cluster 1. Notably, the number of images in cluster 1 is approximately ten times greater than that in cluster 28. This observation highlights a substantial imbalance in the distribution of clusters within the dataset, which should be taken into careful consideration during the model training process. Figure 3 highlights varying dimensions of images in the dataset, emphasising the need for proper resizing to enable efficient feature extraction and representation.

**2.4 Data cleaning and preprocessing.** In order to ensure standardised analysis, we first resize all images to a resolution of 224 x 224 pixels. This step promotes compatibility with pre-trained models and enhances computational efficiency during the training process.

Additionally, we normalise each image by dividing the pixel

**Fig. 2.** Number of images per cluster //// This should be number of images per cluster Can't find the imageeeee, here's a frog

**Fig. 3.** Dimensions of each images in data ///// I couldn't find the full resolution image for the diagram so here's a frog

values by the 99th percentile to minimise any potential variations in pixel intensity and ensure that the images have a standardised range of values.

Furthermore, we apply masking techniques to generate clean images. By removing pixels outside the cell boundaries, we isolate specific cells of interest and reduce noise in the images. This masking process not only improves the interpretability of the images but could also enhance the accuracy of subsequent analysis and model training.

We save the raw and clean images separately with the same hierarchical file structure. This approach maintains the integrity of the original data while providing a cleaner version for further analysis. By preserving this separation, we ensure that preprocessing steps do not impact the original data and can be easily replicated and tracked.

### 2.5 Model development.

***2.5.1 Model and training data design.*** We have chosen two neural architectures, Convolutional Neural Networks (CNNs) and Transformers. CNNs are effective in capturing spatial information and extracting hierarchical features from images, while Transformers excel in modelling global context and dependencies within images.

The raw and clean data are initially divided into train-validation and test sets using an 80:20 split ratio and stratified sampling. However, in this project, we specifically focus on exploring different sampling methods to split train-validation data into 80:20, namely stratified and independent sampling. The inclusion of stratified sampling in our research is crucial to address significant data imbalances between clusters and to examine the impact of a more balanced representation on model performance.

We will develop a total of 8 models to investigate various combinations of architectures, sampling methods, and types of data (raw and clean). This comprehensive approach allows us to explore the effects of these factors on the overall performance of the models.

***2.5.2 Model Building and Training.*** We employed pre-trained models (Resnet50V2 for CNN and Vision Transformers for Transformers) to leverage existing knowledge, improve efficiency and generalisation.

Resnet50V2, with its 50 layers and residual connections, enables successful training of deep networks. In contrast, Vision Transformers utilise transformers to capture global image relationships, facilitating effective feature extraction and pattern recognition.

During training, we utilised a categorical cross-entropy loss function to measure the discrepancy between predicted and actual class labels. We trained the models for 30 epochs, which represents the number of complete passes through the training dataset. Additionally, we selected a batch size of 128, indicating the number of samples processed in each iteration before updating the model's parameters. These choices aimed to balance model convergence and computational time, optimising resource usage while maintaining accurate gradient estimation and model generalisation. We monitored the training process by analysing the accuracy-epochs and loss-epochs plots to assess overfitting and underfitting, an example of which can be found in Appendix A1 .

*CNN* ResNet50V2 has 50 layers and includes convolutional blocks with shortcut connections. These components allow the model to learn intricate representations of input images by capturing important features.

ResNet50V2 addresses the challenge of vanishing gradients in deep networks by using residual blocks with skip connections. These connections enable the model to effectively learn residual mappings, capturing the differences between the input and desired output. This design allows for successful training of deep models by improving information flow.

For our image classification task, we customised ResNet50V2 by adjusting its top layer and adding additional layers. The extracted features from the convolutional blocks are processed through a global average pooling layer to reduce their spatial dimensions. Dense layers with ReLU activation, dropout regularisation, and a final dense layer with softmax activation are applied. These layers generate classification probabilities for the different image classes.

During classification, the network captures low-level features like edges and textures in earlier layers, while deeper layers learn more complex patterns and structures. The global average pooling layer combines the features into a fixed-length representation, and subsequent dense layers interpret this representation to produce class probabilities using softmax activation. The predicted class label is determined based on the highest probability.

*Transformer* In vision transformers, 2D images are transformed into sequences of patches, which are then processed by a transformer architecture. This architecture utilises self-attention mechanisms to capture relationships between patches, enabling the model to extract meaningful representations from the image.

A key feature of vision transformers is the multi-head mechanism, which allows the model to attend to multiple positions or patches simultaneously. Each position has its own attention weights, which are combined to capture a wide range of patterns and relationships within the image. Feed-forward neural networks are also applied to each patch, incorporating non-linear activations to capture complex relationships.

The transformer architecture consists of stacked transformer blocks, connected through attention connections. This allows the model to selectively focus on different parts of the input sequence during generation.

Although primarily used for image classification, vision transformers can generate probabilities for different image classes using a final dense layer with softmax activation. These probabilities facilitate classification based on the highest probability.

***2.5.3 Model Optimisation.*** Each model was optimised until the validation accuracy surpassed the threshold of 15%, which is approximately 5 times the random guessing probability of 1/28. The following optimization techniques were applied.

**2.6 Evaluation.** To evaluate the generalisation capabilities of our models, we used unseen data to simulate real-world scenarios with new and unfamiliar samples. This allowed us to assess the performance of our models in accurately classifying unseen data and determining their ability to generalise beyond the training set.

We selected accuracy as our primary quantitative evaluation metrics for assessing model performance. We chose these metrics due to their simplicity and ease of interpretation. To compare the performance of different models, we utilised side-by-side bar plots to visualise the accuracy values.

As our second evaluation metric, we employed the confusion matrix, which offers valuable insights into the classification performance of our models. This matrix allows us to analyse the accuracy of each cluster, identify misclassifications across different clusters, and assess any imbalances or biases present in the classification results.

We employed innovative evaluation strategies such as Gradient-weighted Class Activation Mapping (Grad-CAM) for CNN and Spatially Adaptive Activation Value(SAAV) for Transformers to enhance

interpretability. These techniques allowed us to visualise where the models focused their attention in the images. By validating the activations, we assessed if the models correctly identified relevant patterns. Deviations from expected patterns indicated potential areas for improvement, such as further training or data collection efforts to enhance pattern understanding (Team, n.d.).

We also evaluate on models storage requirements to optimise resource allocation and minimise the storage resources needed for deploying and running the models, thus enhancing cost-effectiveness.

During the model selection process, we initially prioritised accuracy and the confusion matrix as performance metrics. Additionally, we utilised Grad-CAM and SAAV techniques to assess the interpretability of the models. If a model demonstrated higher interpretability by correctly focusing on the relevant regions of the input data, it was considered a potential candidate, even if its accuracy was slightly lower compared to other models. Furthermore, in cases where there was a significant difference in storage requirements between two architectures, we favoured the architecture with smaller storage needs.

This approach allowed us to choose a final model that struck a balance between satisfactory accuracy, interpretability, and cost-effectiveness. These factors were deemed important for our stakeholders and contributed to the overall success of our methodology.

## 3. Results

### 3.1 Performance metrics.

```
library(ggplot2)

dir <- "Biotechnology/Model_result/"

outputs <- list(
  "output_clean_independent_cnn.csv",
  "output_raw_independent_cnn.csv",
  "output_clean_stratified_cnn.csv",
  "output_raw_stratified_cnn.csv",
  "output_clean_independent_transformers.csv",
  "output_raw_independent_transformers.csv",
  "output_clean_stratified_transformers.csv",
  "output_raw_stratified_transformers.csv"
)

read_csv <- function(f) {
  read.csv(paste0(dir, f))
}

output <- lapply(outputs, read_csv)

accu <- function(o) { unique(o$Test.Accuracy) }

accuracy <- lapply(output, accu)

Test <- data.frame(
  Model = factor(c(
    "Independent/Clean", "Independent/Raw",
    "Stratified/Clean", "Stratified/Raw",
    "Independent/Clean", "Independent/Raw",
    "Stratified/Clean", "Stratified/Raw"
  )),
  Architecture = factor(c(
    "CNN", "CNN", "CNN", "CNN",
    "Tranformers", "Tranformers",
    "Tranformers", "Tranformers"
  )),
  Accuracy = unlist(accuracy)
)

p1 <- ggplot(data = Test) +
  geom_bar(
    aes(x = Model, y = Accuracy,
        fill = Architecture),
    stat = "identity", position = "dodge"
  ) +
  labs(
    title = "Test Accuracy by Model",
    x = "Training Data",
    y = "Accuracy"
  ) +
  theme(axis.text.x =
          element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette = "Set3")

suppressMessages(ggsave("accuracies.pdf", p1))
```

| Architecture | Sampling Method | Data Types | Accuracy (%) |
|---|---|---|---|
| Transformers | Stratified | Clean | 16.78 |
| | | Raw | 15.25 |
| | Independent | Clean | 16.49 |
| | | Raw | 15.64 |
| CNN | Stratified | Clean | 18.22 |
| | | Raw | 16.32 |
| | Independent | Clean | 18.61 |
| | | Raw | 17.69 |

**Table 1. Accuracy comparisons between models**

This *pinp is not PNAS* template started when the introduction to Rcpp by Eddelbuettel and Balamuta (2017) was converted into this updated Rcpp Introduction vignette. It is based on the pnas_article template of the wonderful rticles package by Allaire *et al.* (2017b). The conversion from markdown to latex is facilitated by rmarkdown (Allaire *et al.*, 2017a) and knitr (Xie, 2017). The underlying LaTeX macros are from pnas.org.

The remainder of the document carries over from the corresponding pnas_article template document. but has been edited and updated to our use case. A few specific tips follow. In general, for fine-tuning some knowledge of LaTeX is helpful.

**Author Affiliations.** Per common academic best practice, you can include your department, institution, and complete address, with the ZIP/postal code, for each author. Use lower case letters to match authors with institutions, as shown in the example. Authors with an ORCID ID may supply this information at submission.

**Document Options.** We support several options via the YAML header

- Setting a DOI or URL footer, for example for the CRAN package URL, which is placed in the bottom-left footer of the title page and even pages;
- Setting a footer label, for example *YourPackage Vignette* stating your package, which is placed in the bottom-right footer on odd pages;

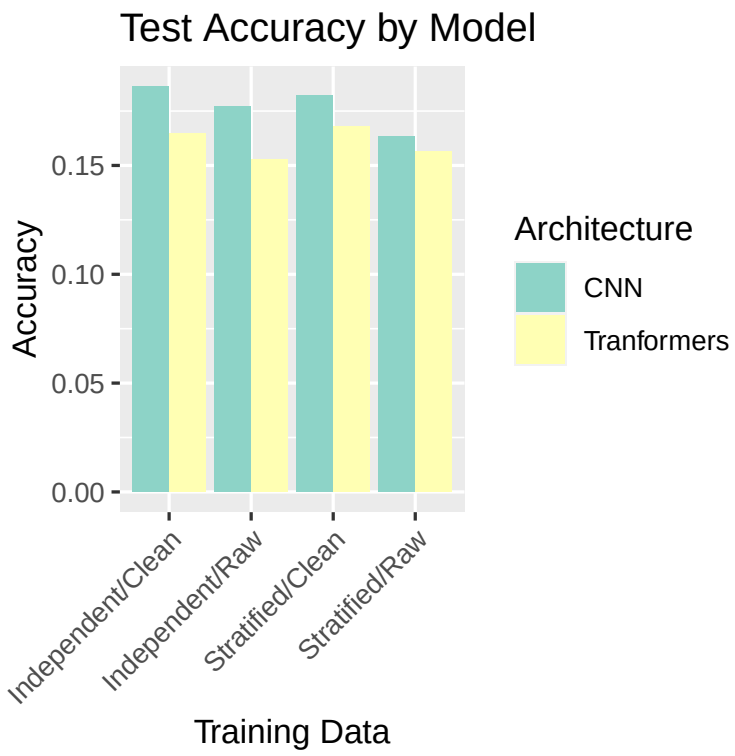


**Fig. 4.** Wide ggplot2 figure

- Setting a free-form author field used on the inside footer;
- Optional *Draft* watermark to be added to each page;
- Line of custom text in subtitle (`date_subtitle`) suitable to give publication info of the draft, e.g. journal name in a post-print;
- Document date that appears in the footer can be specified manually using `document_date`.

**References.** Here we differ from PNAS and suggest natbib. References will appear in author-year form. Use `\citet{}`, `\citep{}`, etc as usual.

We default to the `jss.bst` style. To switch to a different bibliography style, please use `biblio-style: style` in the YAML header.

**Inline R Code.** The PNAS sample included a fixed PNG image here, but this document prefers to show the results and embedding of *R* code.

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
    geom_point(size=3, aes(colour=factor(cyl))) +
    theme(legend.position="none")
```

Here we use a standard knitr bloc with explicit options for

- figure width and height (`fig.width`, `fig.height`), both set to three inches;
- whether the code is shown (`echo=TRUE`); and
- the caption (`fig.cap`) as shown above.

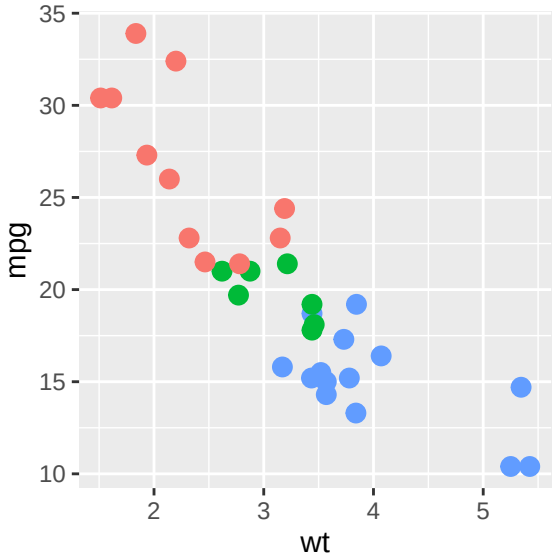


**Fig. 5.** Narrow ggplot2 figure

**Single column equations.** Authors may use 1- or 2-column equations in their article, according to their preference.

To allow an equation to span both columns, options are to use the `\begin{figure*}...\end{figure*}` environment mentioned above for figures. The `\begin{widetext}...\end{widetext}` environment as shown in equation 1 below is deprecated, but LaTeX commands `\onecolumn` and `\twocolumn` work fine.

Please note that this option may run into problems with floats and footnotes, as mentioned in the cuted package documentation. In the case of problems with footnotes, it may be possible to correct the situation using commands `\footnotemark` and `\footnotetext`.

$$\begin{aligned}(x+y)^3 &= (x+y)(x+y)^2 \\ &= (x+y)(x^2+2xy+y^2) \\ &= x^3+3x^2y+3xy^3+x^3.\end{aligned} \tag{1}$$

## Appendix

```
history <- read.csv(
  "Biotechnology/History/c_i_cnn.csv"
)

library(ggplot2)

custom_colors <- c("blue",
                   "red")
custom_labels <- c("Train",
                   "Validation")

ggplot(data = history) +
  geom_line(aes(x = epoch,
                y = train_accuracy,
                color = "Train")) +
  geom_line(aes(x = epoch,
                y = validation_accuracy,
                color = "Validation")) +
  scale_color_manual(values = custom_colors,
                     labels = custom_labels) +
  labs(x = "Epoch",
       y = "Accuracy",
       colour = "") +
  theme_bw()
```

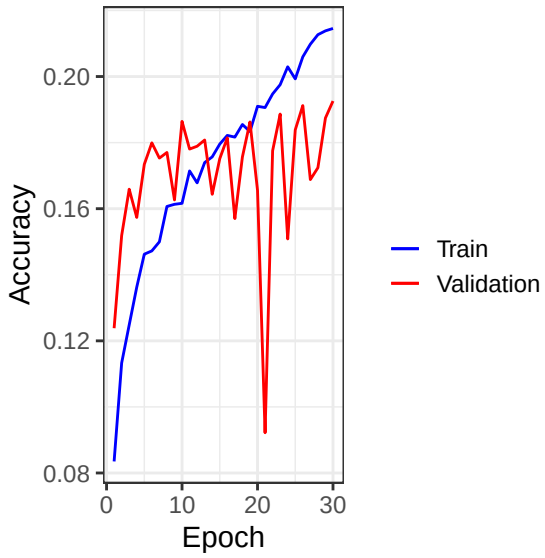


**Fig. 6.** Model accuracy for CNN Independent/clean

```
ggplot(data = history) +
  geom_line(aes(x = epoch,
                y = loss,
                color = "Train")) +
  geom_line(aes(x = epoch,
                y = validation_loss,
                color = "Validation")) +
  scale_color_manual(values = custom_colors,
                     labels = custom_labels) +
  labs(x = "Epoch",
       y = "Loss",
       title = "Model Loss",
       colour = "") +
  theme_bw()
```

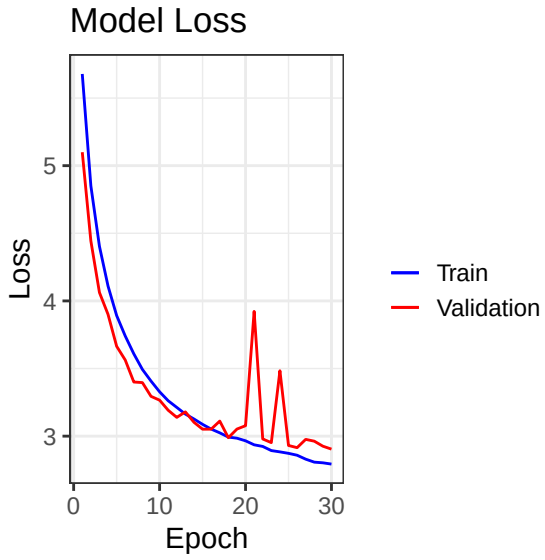


**Fig. 7.** Model loss for CNN Independent/clean

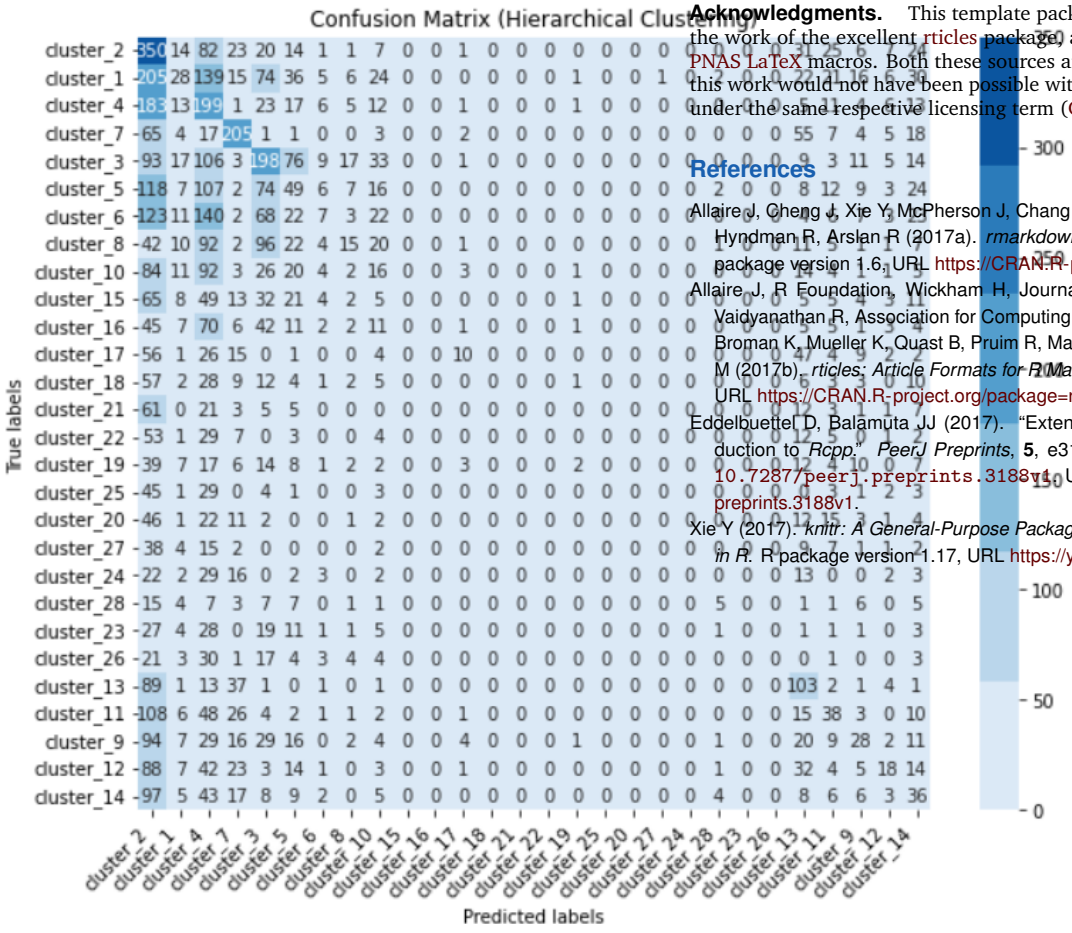


**Fig. 8.** Flower one.

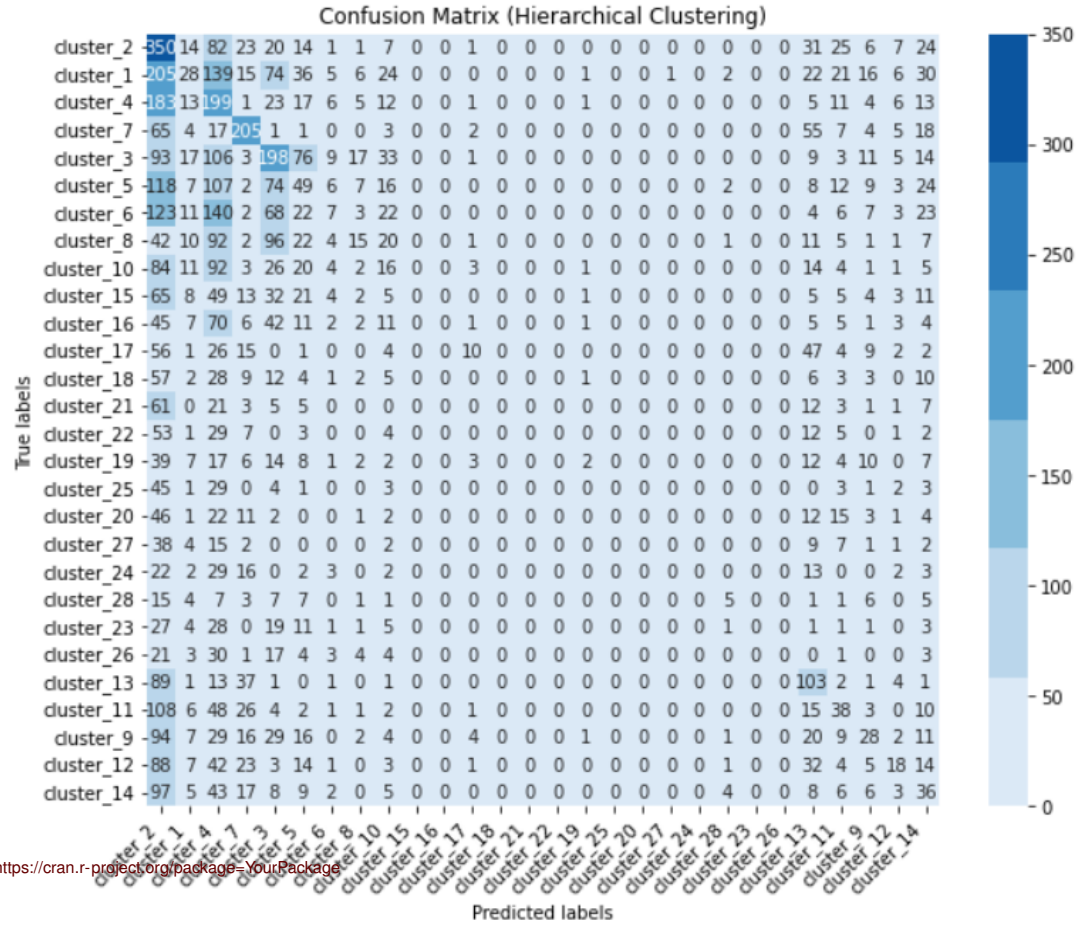


**Fig. 9.** Flower two.

**Acknowledgments.** This template package builds upon, and extends, the work of the excellent rticles package, and both packages rely on the PNAS LaTeX macros. Both these sources are gratefully acknowledged as this work would not have been possible without them. Our extensions are under the same respective licensing term (GPL-3 and LPPL (>= 1.3)).

## References

Allaire J, Cheng J, Xie Y, McPherson J, Chang W, Allen J, Wickham H, Atkins A, Hyndman R, Arslan R (2017a). *rmarkdown: Dynamic Documents for R*. R package version 1.6, URL https://CRAN.R-project.org/package=rmarkdown.

Allaire J, R Foundation, Wickham H, Journal of Statistical Software, Xie Y, Vaidyanathan R, Association for Computing Machinery, Boettiger C, Elsevier, Broman K, Mueller K, Quast B, Pruim R, Marwick B, Wickham C, Keyes O, Yu M (2017b). *rticles: Article Formats for R Markdown*. R package version 0.4.1, URL https://CRAN.R-project.org/package=rticles.

Eddelbuettel D, Balamuta JJ (2017). "Extending *R* with *C++*: A Brief Introduction to *Rcpp*." *PeerJ Preprints*, **5**, e3188v1. ISSN 2167-9843. doi:10.7287/peerj.preprints.3188v1, URL https://doi.org/10.7287/peerj.preprints.3188v1.

Xie Y (2017). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.17, URL https://yihui.name/knitr/.